NAME

transchan — command handler API of channel transforms

SYNOPSIS

cmdPrefix *option* ?*arg arg ...*?

DESCRIPTION

The Tcl-level handler for a channel transformation has to be a command with subcommands (termed an *ensemble* despite not implying that it must be created with **namespace ensemble create**; this mechanism is not tied to **namespace ensemble** in any way). Note that *cmdPrefix* is whatever was specified in the call to **chan push**, and may consist of multiple arguments; this will be expanded to multiple words in place of the prefix.

Of all the possible subcommands, the handler *must* support **initialize** and **finalize**. Transformations for writable channels must also support **write**, and transformations for readable channels must also support **read**.

Note that in the descriptions below *cmdPrefix* may be more than one word, and *handle* is the value returned by the **chan push** call used to create the transformation.

GENERIC SUBCOMMANDS

The following subcommands are relevant to all types of channel.

cmdPrefix **clear** *handle*
: This optional subcommand is called to signify to the transformation that any data stored in internal buffers (either incoming or outgoing) must be cleared. It is called when a **chan seek** is performed on the channel being transformed.

cmdPrefix **finalize** *handle*
: This mandatory subcommand is called last for the given *handle*, and then never again, and it exists to allow for cleaning up any Tcl-level data structures associated with the transformation. *Warning!* Any errors thrown by this subcommand will be ignored. It is not guaranteed to be called if the interpreter is deleted.

cmdPrefix **initialize** *handle mode*
: This mandatory subcommand is called first, and then never again (for the given *handle*). Its responsibility is to initialize all parts of the transformation at the Tcl level. The *mode* is a list containing any of **read** and **write**.

  write
  : implies that the channel is writable.

  read
  : implies that the channel is readable.

  The return value of the subcommand should be a list containing the names of all subcommands supported by this handler. Any error thrown by the subcommand will prevent the creation of the transformation. The thrown error will appear as error thrown by **chan push**.

READ-RELATED SUBCOMMANDS

These subcommands are used for handling transformations applied to readable channels; though strictly **read** is optional, it must be supported if any of the others is or the channel will be made non-readable.

cmdPrefix **drain** *handle*
: This optional subcommand is called whenever data in the transformation input (i.e. read) buffer has to be forced upward, i.e. towards the user or script. The result returned by the method is taken as the *binary* data to push upward to the level above this transformation (the reader or a higher-level transformation).

  In other words, when this method is called the transformation cannot defer the actual transformation operation anymore and has to transform all data waiting in its internal read buffers and return the result of that action.

cmdPrefix **limit?** *handle*
: This optional subcommand is called to allow the Tcl I/O engine to determine how far ahead it should read. If present, it should return an integer number greater than zero which indicates how many bytes ahead should be read, or an integer less than zero to indicate that the I/O engine may read as far ahead as it likes.

cmdPrefix **read** *handle buffer*
: This subcommand, which must be present if the transformation is to work with readable channels, is called whenever the base channel, or a transformation below this transformation, pushes data upward. The *buffer* contains the binary data which has been given to us from below. It is the responsibility of this subcommand to actually transform the data. The result returned by the subcommand is taken as the binary data to push further upward to the transformation above this transformation. This can also be the user or script that originally read from the channel.

  Note that the result is allowed to be empty, or even less than the data we received; the transformation is not required to transform everything given to it right now. It is allowed to store incoming data in internal buffers and to defer the actual transformation until it has more data.

WRITE-RELATED SUBCOMMANDS

These subcommands are used for handling transformations applied to writable channels; though strictly **write** is optional, it must be supported if any of the others is or the channel will be made non-writable.

cmdPrefix **flush** *handle*
: This optional subcommand is called whenever data in the transformation 'write' buffer has to be forced downward, i.e. towards the base channel. The result returned by the subcommand is taken as the binary data to write to the transformation below the current transformation. This can be the base channel as well.

  In other words, when this subcommand is called the transformation cannot defer the actual transformation operation anymore and has to transform all data waiting in its internal write buffers and return the result of that action.

cmdPrefix **write** *handle buffer*
: This subcommand, which must be present if the transformation is to work with writable channels, is called whenever the user, or a transformation above this transformation, writes data downward. The *buffer* contains the binary data which has been written to us. It is the responsibility of this subcommand to actually transform the data.

The result returned by the subcommand is taken as the binary data to write to the transformation below this transformation. This can be the base channel as well. Note that the result is allowed to be empty, or less than the data we got; the transformation is not required to transform everything which was written to it right now. It is allowed to store this data in internal buffers and to defer the actual transformation until it has more data.

SEE ALSO

chan, **refchan**

KEYWORDS

API, channel, ensemble, prefix, transformation